

MPI collective communication

10/1/07

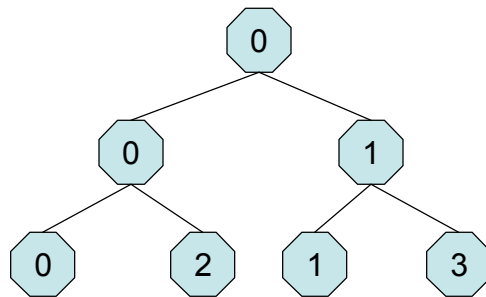
Tree-structured communication

- Suppose we would like to broadcast specific data from one processor to the remaining $p - 1$ processors.
- As mentioned before, this can be done using a tree structure.

Basic framework

- Given that we have p processors, our tree structure will have p leaves.
- Each processor will be involved in at most $\log(p) + 1$ stages.

Organization



Advantages

- By structuring our communication as illustrated before, there is only one communication per step.
- For example if $p = 8$:
 - Stage 1: 0 sends to 1
 - Stage 2: 0 sends to 2; 1 sends to 3
 - Stage 3: 0 sends to 4; 1 sends to 5; 2 sends to 6; 3 sends to 7

Generalization

- If $2^{\text{stage}} \leq \text{my_rank} \leq 2^{\text{stage}+1}$
 - Receive from $\text{my_rank} - 2^{\text{stage}}$
- Otherwise if $\text{my_rank} < 2^{\text{stage}}$
 - Send to $\text{my_rank} + 2^{\text{stage}}$

Our code vs. MPI

- Based on the programming assignment, we now know how to use MPI_Send and MPI_Recv.
- Suppose we implement our communication strategy using a loop and the preceding observations. Is this better than MPI?

Pacheco's implementation

- Pp 67-71; v2 is the MPI-based solution

P	<i>nCUBE2</i>		<i>Paragon</i>		<i>SP2</i>	
	<i>v1</i>	<i>v2</i>	<i>v1</i>	<i>v2</i>	<i>v1</i>	<i>v2</i>
2	0.59	0.69	0.21	0.43	0.15	0.16
8	4.7	1.9	0.84	0.93	0.55	0.35
32	19.0	3.0	3.2	1.3	2.0	0.57

Reduction

- Collective communication called reductions are the same as the ones we discussed related to parallel prefix.
- These are also called *scan* in the terminology.

Usage

- `int MPI_Reduce (`
 - `void* operand,`
 - `void* result, // stored on root processor`
 - `int count,`
 - `MPI_datatype datatype,`
 - `MPI_Op operator,`
 - `int root,`
 - `MPI_Comm comm)`

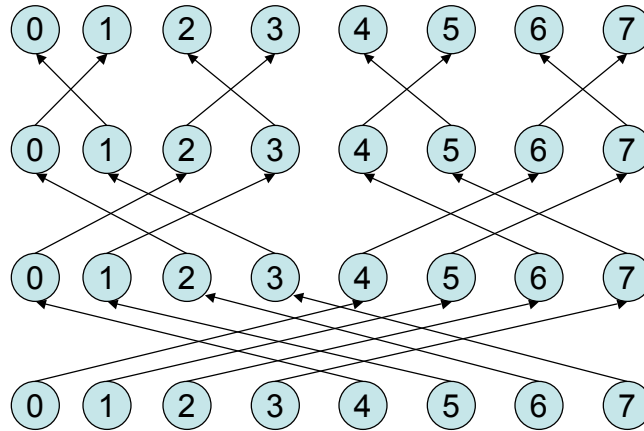
Predefined operators

Operator	Function
MPI_MAX	Maximum
MPI_SUM	Sum
MPI LAND	Logical AND
MPI_PROD	Product
MPI_MAXLOC	Maximum and location
MPI_MINLOC	Minimum and location

Allreduce

- In class, we discussed performing parallel prefix using hypercubic permutations.
- Today, we discussed tree-based communications for broadcast.
- Can we perform a tree-structured reduce rooted at all of the processors simultaneously?

Butterfly



Discussion

- Suppose we add another line connecting a processor to itself in adjacent rows in the preceding slide.
- The result is a tree rooted at each process!

Usage

- int MPI_AllReduce (
 - void* operand,
 - void* result, // stored on all processors
 - int count,
 - MPI_datatype datatype,
 - MPI_Op operator,
 - MPI_Comm comm)

Tags and collective communication

- Note that none of the collective communications involve tags, as the communication in your assignment did.
- This is because they all should be *synchronous* operations, or one that can not complete until all others have started.
- The implication is all processors must execute the same operation in the same order.

Parallel programming

- Pacheco discusses parallel code development in chapter 9.
- For serial programs, we typically do the following:
 - Examine the source code
 - Add debugging output statements
 - Use a symbolic debugger

Parallel debugging

- At this time, an affordable debugger is not readily available to the community.
- This is further complicated by nondeterminism and architecture
 - It is not uncommon for software to work on a cluster and not work on another system

Common bugs in parallel code

- Trying to receive before sending, or trying to receive when there is no send.
 - Called “deadlock”
- Incorrect parameters to send/receive
- Serial bugs

Helpful hints

- Ensure communications are behaving using IO messages
 - Use fflush to clear the buffer as needed
- Start small where possible.
 - Serial before parallel
 - 2 processors before 16